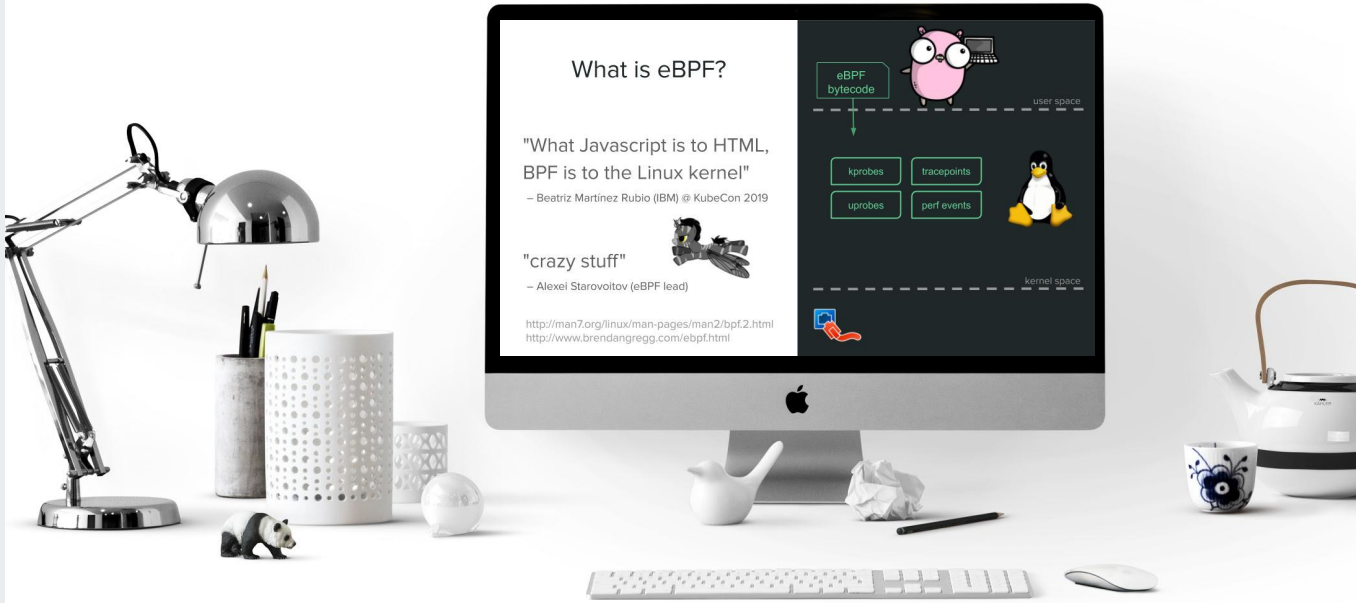
# eBPF and Go

Florian Lehner
XVII.  Zürich Go Meetup

# XIV. Zürich Go Meetup




https://www.pexels.com/photo/silver-imac-near-white-ceramic-kettle-205316/

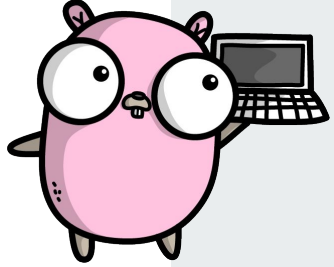
# classic BPF

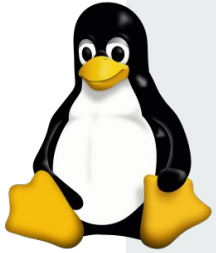
- two 32 bit registers
- SO_ATTACH_FILTER

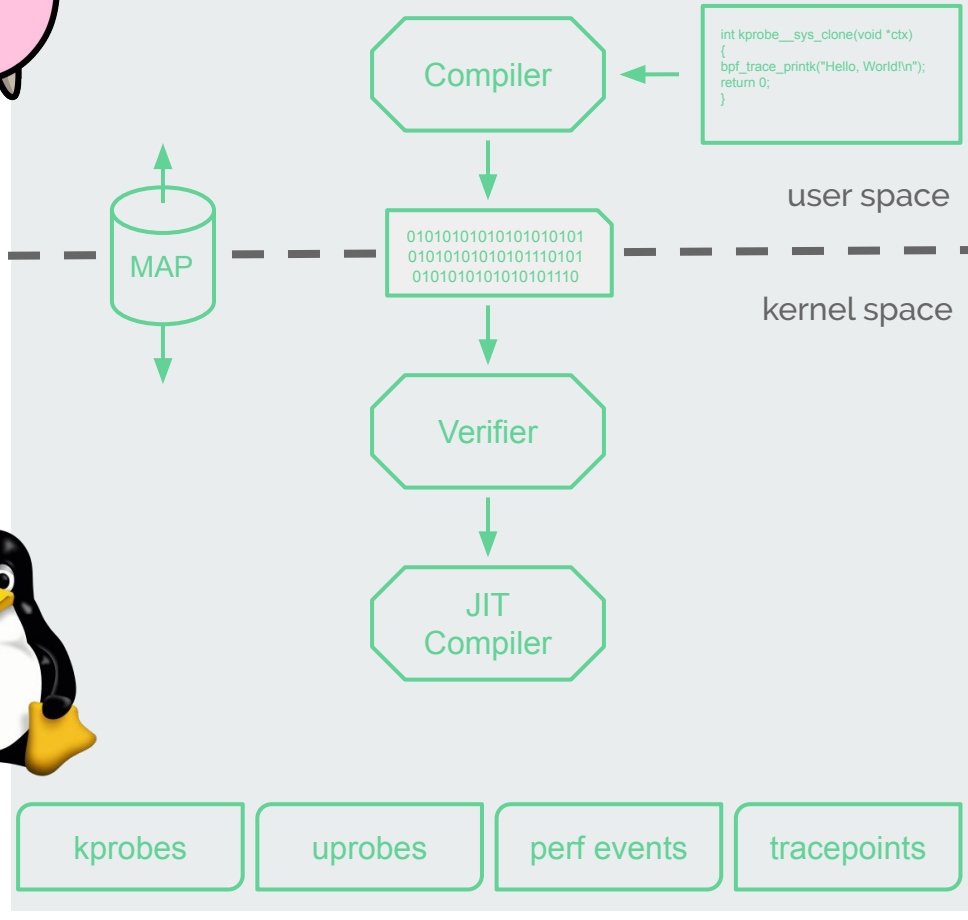
```
# tcpdump -i lo -d tcp and port 22
(000) ldh      [12]
(001) jeq      #0x86dd          jt 2      jf 8
(002) ldb      [20]
(003) jeq      #0x6             jt 4 jf 19
(004) ldh      [54]
(005) jeq      #0x16            jt 18     jf 6
(006) ldh      [56]
(007) jeq      #0x16            jt 18     jf 19
(008) jeq      #0x800           jt 9      jf 19
(009) ldb      [23]
(010) jeq      #0x6             jt 11     jf 19
(011) ldh      [20]
(012) jset     #0x1fff          jt 19     jf 13
(013) ldxb     4*([14]&0xf)
(014) ldh      [x + 14]
(015) jeq      #0x16            jt 18     jf 16
(016) ldh      [x + 16]
(017) jeq      #0x16            jt 18     jf 19
(018) ret      #262144
(019) ret      #0
```

# eBPF

- eleven 64 bit registers
- 512 byte stack
- kernel side helper functions
- maximum of 1 million instruction
- maps for data exchange
- [man 2 bpf]







# Comparison

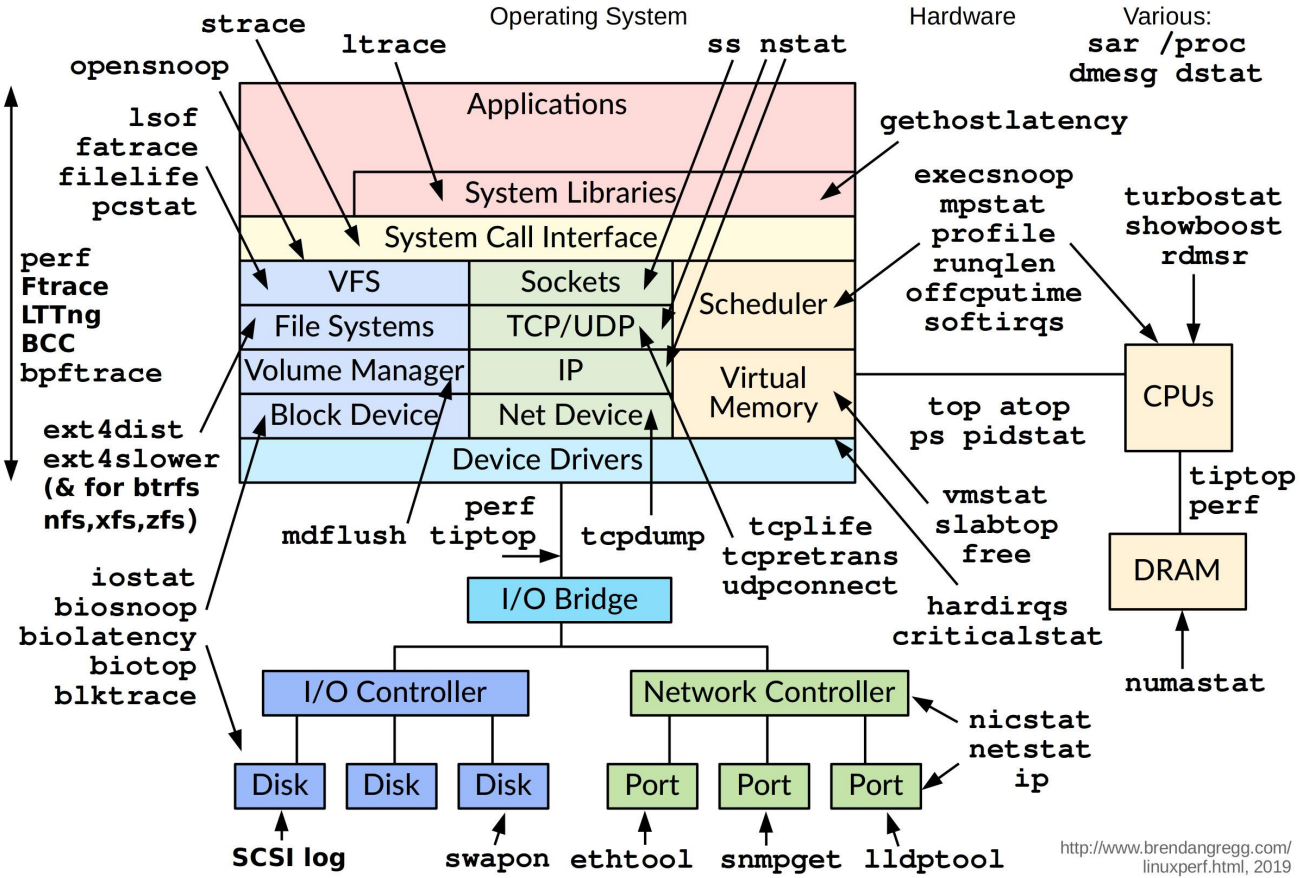
## eBPF

- eleven 64 bit registers
- 512 byte stack
- kernel side helper functions
- maximum of 1 million instruction per program
- maps for data exchange
- man 2 bpf

## classic BPF

- two 32 bit registers
- SO_ATTACH_FILTER

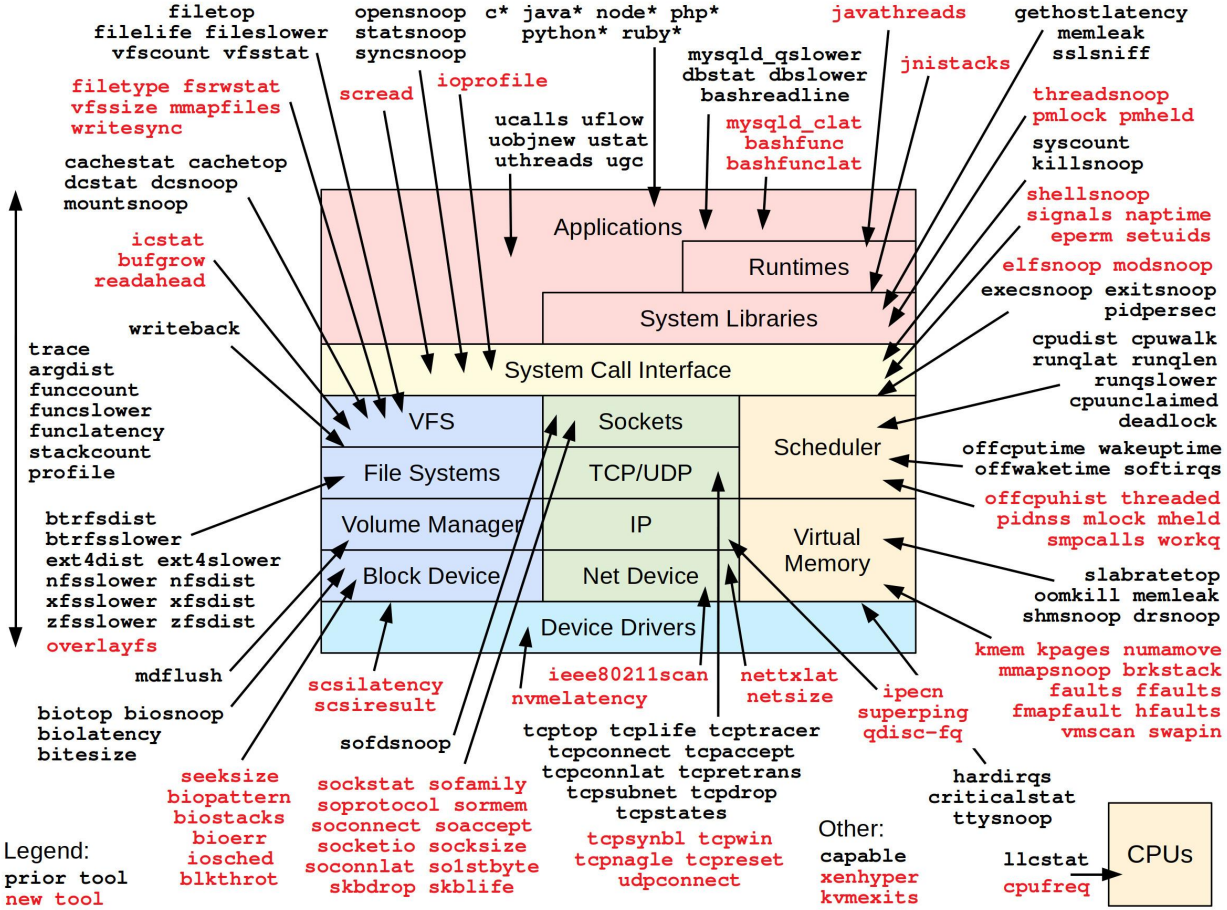
# Linux Performance Observability Tools




http://www.brendangregg.com/linuxperf.html, 2019

http://www.brendangregg.com/linuxperf.html

**New tools** developed for the book **BPF Performance Tools: Linux System and Application Observability**
by Brendan Gregg (Addison Wesley, 2019), which also covers **prior BPF tools**



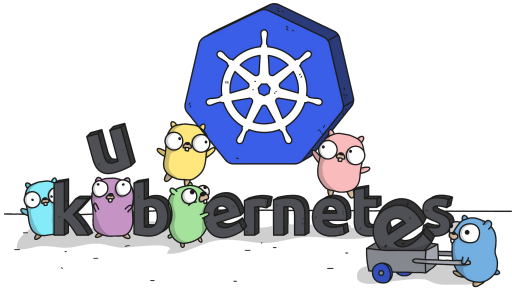

http://www.brendangregg.com/linuxperf.html

# Where do I find BPF?

```
struct bpf_insn insn[] = {
   BPF_JMP_IMM(BPF_JNE, BPF_REG_7, htobe16(protocol), 0),

   BPF_MOV64_REG(BPF_REG_1, BPF_REG_6),
   BPF_MOV32_IMM(BPF_REG_2, addr_offset),

   BPF_MOV64_REG(BPF_REG_3, BPF_REG_10),
   BPF_ALU64_IMM(BPF_ADD, BPF_REG_3, -addr_size),

   BPF_MOV32_IMM(BPF_REG_4, addr_size),
   BPF_RAW_INSN(BPF_JMP | BPF_CALL, 0, 0, 0, BPF_FUNC_skb_load_bytes),

   BPF_LD_MAP_FD(BPF_REG_1, map_fd),
   BPF_MOV64_REG(BPF_REG_2, BPF_REG_10),
   BPF_ALU64_IMM(BPF_ADD, BPF_REG_2, -addr_size - sizeof(uint32_t)),
   BPF_ST_MEM(BPF_W, BPF_REG_2, 0, addr_size * 8),

   BPF_RAW_INSN(BPF_JMP | BPF_CALL, 0, 0, 0,
BPF_FUNC_map_lookup_elem),
   BPF_JMP_IMM(BPF_JEQ, BPF_REG_0, 0, 1),
   BPF_ALU32_IMM(BPF_OR, BPF_REG_8, verdict),
};
```

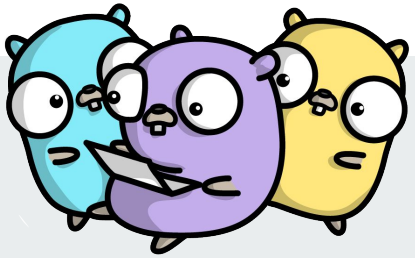
[ ● ◀ ] systemd





# Writing eBPF in Go

# Implementations for



| | Type | Base | Functions |
|---|---|---|---|
| golang.org/x/net/bpf | classic BPF | Go | - |
| github.com/iovisor/gobpf<br>github.com/dropbox/goebpf | eBPF | C and Go | + |
| github.com/cilium/ebpf | eBPF | Go | + |

```
import "C"

// Write your eBPF module in C
const source string = `
#include <uapi/linux/bpf.h>

int tcExample(...) {
        ...
}
`

// Create an eBPF module
module := bpf.NewModule(...)

// Open a netlink socket
rtnl, err := tc.Open(...)

// Add a queueing discipline
rtnl.Qdisc().Add(...)

// Add filter with the eBPF module
rtnl.Filter().Add(...)

for {
        // handle data from the eBPF module
        data := <-channel
}
```

https://github.com/florianl/monitoringIPbasedNetworks/blob/master/eBPF/trafficControl/main.go

```
insns := asm.Instructions{
        // r1 has ctx
        // r0 = ctx[16] (aka protocol)
        asm.LoadMem(asm.R0, asm.R1, 16, asm.Word),

        // Perhaps ipv6
        asm.LoadImm(asm.R2, int64(ETH_P_IPV6), asm.DWord),
        asm.HostTo(asm.BE, asm.R2, asm.Half),
        asm.JEq.Reg(asm.R0, asm.R2, "ipv6"),
...
        asm.Return(),
}

prog, err := ebpf.NewProgram(&ebpf.ProgramSpec{
        Name:        "distance_filter",
        Type:        ebpf.SocketFilter,
        License:     "GPL",
        Instructions: insns,
})
```

https://github.com/cilium/ebpf/blob/master/example_sock_extract_dist_test.go
https://blog.cloudflare.com/epbf_sockets_hop_distance/

```
import "C"

// Write your eBPF module in C
const source string = `
#include <uapi/linux/bpf.h>

int tcExample(...) {
        ...
}
`

// Create an eBPF module
module := bpf.NewModule(...)

// Open a netlink socket
rtnl, err := tc.Open(...)

// Add a queueing discipline
rtnl.Qdisc().Add(...)

// Add filter with the eBPF module
rtnl.Filter().Add(...)

for {
        // handle data from the eBPF module
        data := <-channel
}
```

https://github.com/florianl/monitoringIPbasedNetworks/blob/master/eBPF/trafficControl/main.go

```
insns := asm.Instructions{
        // r1 has ctx
        // r0 = ctx[16] (aka protocol)
        asm.LoadMem(asm.R0, asm.R1, 16, asm.Word),

        // Perhaps ipv6
        asm.LoadImm(asm.R2, int64(ETH_P_IPV6), asm.DWord),
        asm.HostTo(asm.BE, asm.R2, asm.Half),
        asm.JEq.Reg(asm.R0, asm.R2, "ipv6"),
...
        asm.Return(),
}

prog, err := ebpf.NewProgram(&ebpf.ProgramSpec{
        Name:        "distance_filter",
        Type:        ebpf.SocketFilter,
        License:     "GPL",
        Instructions: insns,
})
```

https://github.com/cilium/ebpf/blob/master/example_sock_extract_dist_test.go
https://blog.cloudflare.com/epbf_sockets_hop_distance/

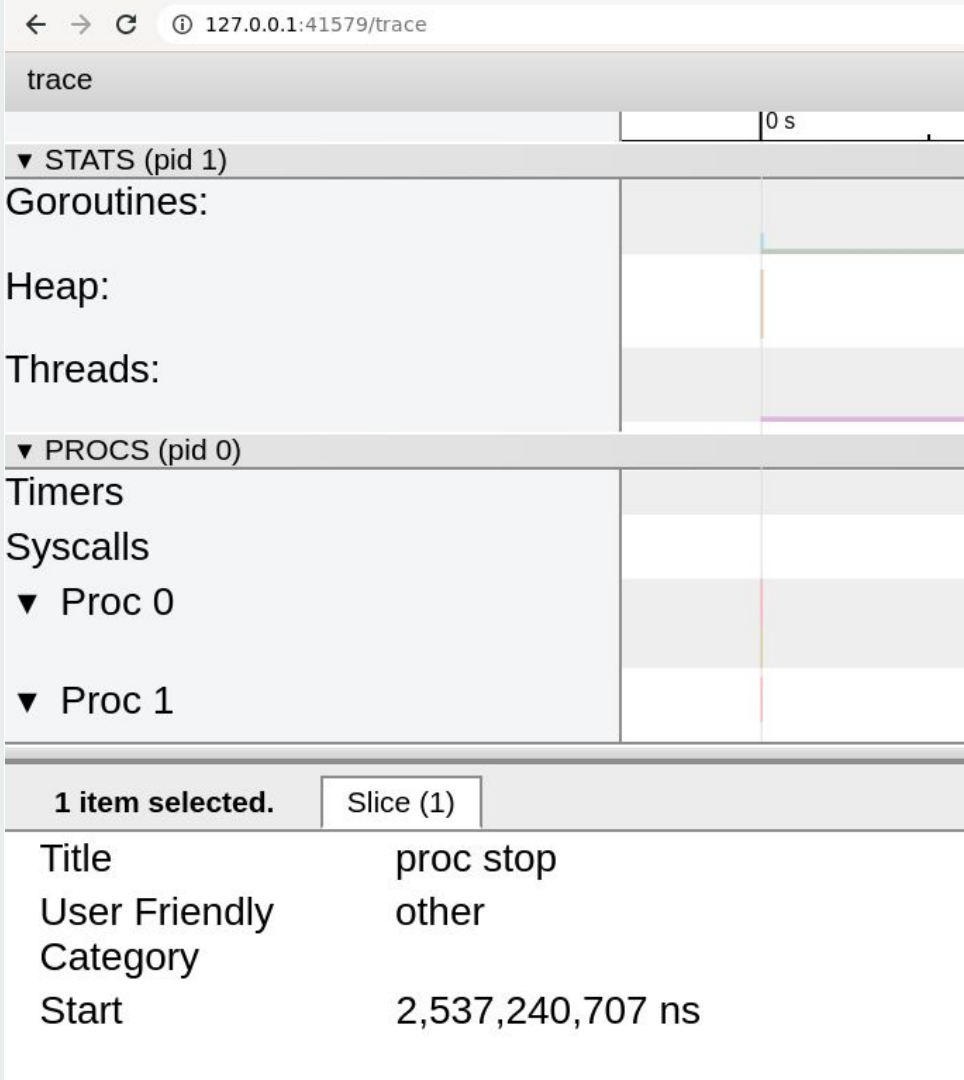
# Inspecting Go with eBPF

```
//go:noinline
func fibonacciRecursive(n uint32) uint32 {
	if n < 2 {
		return n
	}
	return fibonacciRecursive(n-1) + fibonacciRecursive(n-2)
}

//go:noinline
func fibonacciCache(n uint32, cache map[uint32]uint32) uint32 {
	var i uint32
	for i = 2; i <= n; i++ {
		tmp := cache[i-1] + cache[i-2]
		cache[i] = tmp
	}
	return cache[n]
}

//go:noinline
func fibonacciLoop(n uint32) uint32 {
	var a uint32 = 0
	var b uint32 = 1
	for i := 0; i < int(n); i++ {
		a, b = b, a+b
	}
	return a
}
```

https://gist.github.com/florianl/1080769dd6dee231cd21ac6a240f45b5

```
//go:noinline
func fibonacciRecursive(n uint32) uint32 {
	if n < 2 {
		return n
	}
	return fibonacciRecursive(n-1) + fibonacciRecursive(n-2)
}

//go:noinline
func fibonacciCache(n uint32, cache map[uint32]uint32) uint32 {
	var i uint32
	for i = 2; i <= n; i++ {
		tmp := cache[i-1] + cache[i-2]
		cache[i] = tmp
	}
	return cache[n]
}

//go:noinline
func fibonacciLoop(n uint32) uint32 {
	var a uint32 = 0
	var b uint32 = 1
	for i := 0; i < int(n); i++ {
		a, b = b, a+b
	}
	return a
}
```

https://gist.github.com/florianl/1080769dd6dee231cd21ac6a240f45b5



```
//go:noinline
func fibonacciRecursive(n uint32) uint32 {
	if n < 2 {
		return n
	}
	return fibonacciRecursive(n-1) + fibonacciRecursive(n-2)
}

//go:noinline
func fibonacciCache(n uint32, cache map[uint32]uint32) uint32 {
	var i uint32
	for i = 2; i <= n; i++ {
		tmp := cache[i-1] + cache[i-2]
		cache[i] = tmp
	}
	return cache[n]
}

//go:noinline
func fibonacciLoop(n uint32) uint32 {
	var a uint32 = 0
	var b uint32 = 1
	for i := 0; i < int(n); i++ {
		a, b = b, a+b
	}
	return a
}
```

https://gist.github.com/florianl/1080769dd6dee231cd21ac6a240f45b5

```
# bpftrace -l 'uprobe:/tmp/fibonacci:main.*'
uprobe:/tmp/fibonacci:main.fibonacciCache
uprobe:/tmp/fibonacci:main.fibonacciLoop
uprobe:/tmp/fibonacci:main.fibonacciRecursive
uprobe:/tmp/fibonacci:main.init.0
uprobe:/tmp/fibonacci:main.main
```

https://github.com/iovisor/bpftrace

```
//go:noinline
func fibonacciRecursive(n uint32) uint32 {
	if n < 2 {
		return n
	}
	return fibonacciRecursive(n-1) + fibonacciRecursive(n-2)
}

//go:noinline
func fibonacciCache(n uint32, cache map[uint32]uint32) uint32 {
	var i uint32
	for i = 2; i <= n; i++ {
		tmp := cache[i-1] + cache[i-2]
		cache[i] = tmp
	}
	return cache[n]
}

//go:noinline
func fibonacciLoop(n uint32) uint32 {
	var a uint32 = 0
	var b uint32 = 1
	for i := 0; i < int(n); i++ {
		a, b = b, a+b
	}
	return a
}
```

https://gist.github.com/florianl/1080769dd6dee231cd21ac6a240f45b5

```
# bpftrace -e
'uprobe:/tmp/fibonacci:main.fibonacciLoop
{
  printf("arg: %d\n", sarg0);
}'

arg: 9
arg: 12
arg: 7
arg: 29
arg: 12
arg: 18
arg: 15
...
```

https://github.com/iovisor/bpftrace

```
//go:noinline
func fibonacciRecursive(n uint32) uint32 {
	if n < 2 {
		return n
	}
	return fibonacciRecursive(n-1) + fibonacciRecursive(n-2)
}

//go:noinline
func fibonacciCache(n uint32, cache map[uint32]uint32) uint32 {
	var i uint32
	for i = 2; i <= n; i++ {
		tmp := cache[i-1] + cache[i-2]
		cache[i] = tmp
	}
	return cache[n]
}

//go:noinline
func fibonacciLoop(n uint32) uint32 {
	var a uint32 = 0
	var b uint32 = 1
	for i := 0; i < int(n); i++ {
		a, b = b, a+b
	}
	return a
}
```

https://gist.github.com/florianl/1080769dd6dee231cd21ac6a240f45b5

```
# bpftrace -e
'uprobe:/tmp/fibonacci:main.fibonacciCache
{
  @[ustack] = count()
}'

@[
  main.fibonacciCache+0
   runtime.main+542
   0xc00007c000
   0x89481eebc0313474
]: 1
```

https://github.com/iovisor/bpftrace

```
//go:noinline
func fibonacciRecursive(n uint32) uint32 {
        if n < 2 {
                return n
        }
        return fibonacciRecursive(n-1) + fibonacciRecursive(n-2)
}

//go:noinline
func fibonacciCache(n uint32, cache map[uint32]uint32) uint32 {
        var i uint32
        for i = 2; i <= n; i++ {
                tmp := cache[i-1] + cache[i-2]
                cache[i] = tmp
        }
        return cache[n]
}

//go:noinline
func fibonacciLoop(n uint32) uint32 {
        var a uint32 = 0
        var b uint32 = 1
        for i := 0; i < int(n); i++ {
                a, b = b, a+b
        }
        return a
}
```

https://gist.github.com/florianl/1080769dd6dee231cd21ac6a240f45b5

```
# bpftrace -e
'uprobe:/tmp/fibonacci:main.fibonacciRecursive
{
  @start[pid] = nsecs;
}
uretprobe:/tmp/fibonacci:main.fibonacciRecursive
/@start[pid]/
{
  @ns[comm] = hist(nsecs - @start[pid]);
  delete(@start[pid]);
}'
@ns[fibonacci]:
[2K, 4K)          973 |@@@@@@@@@@@@@@@@@@@@@@@@@@|
[4K, 8K)          326 |@@@@@@@@@                   |
[8K, 16K)          27 |@                           |
[16K, 32K)          2 |                            |
[32K, 64K)          1 |                            |
[64K, 128K)         1 |                            |
```

https://github.com/iovisor/bpftrace

```
//go:noinline
func fibonacciRecursive(n uint32) uint32 {
	if n < 2 {
		return n
	}
	return fibonacciRecursive(n-1) + fibonacciRecursive(n-2)
}

//go:noinline
func fibonacciCache(n uint32, cache map[uint32]uint32) uint32 {
	var i uint32
	for i = 2; i <= n; i++ {
		tmp := cache[i-1] + cache[i-2]
		cache[i] = tmp
	}
	return cache[n]
}

//go:noinline
func fibonacciLoop(n uint32) uint32 {
	var a uint32 = 0
	var b uint32 = 1
	for i := 0; i < int(n); i++ {
		a, b = b, a+b
	}
	return a
}
```

https://gist.github.com/florianl/1080769dd6dee231cd21ac6a240f45b5

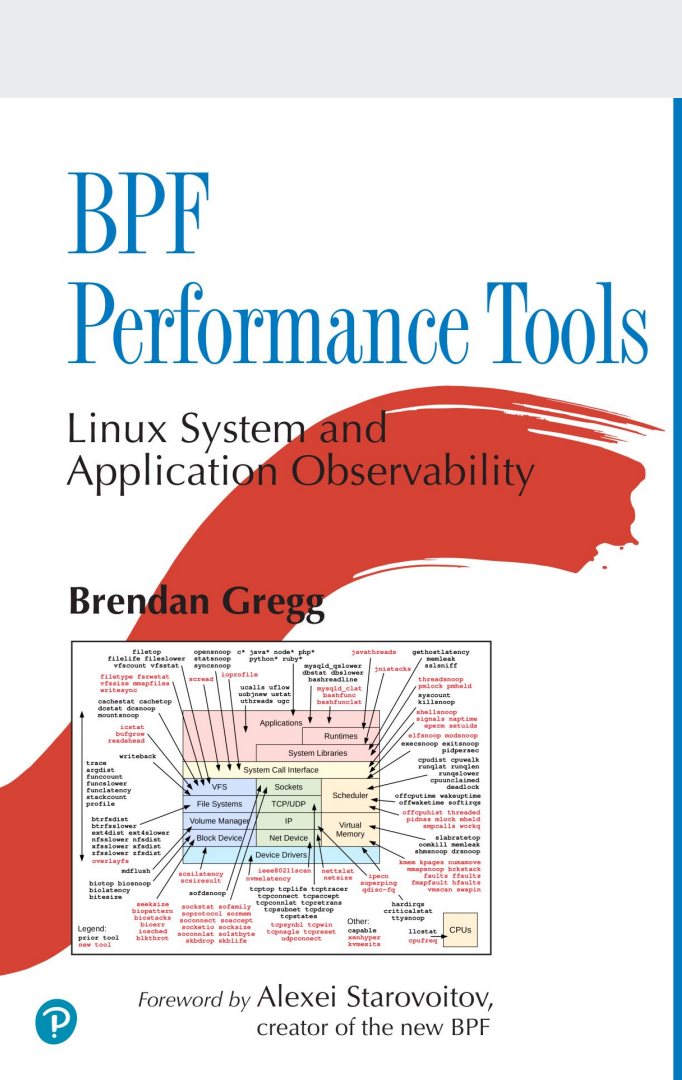
```
# bpftrace -e
'uprobe:/tmp/fibonacci:main.fibonacciLoop
{
  @start[pid] = nsecs;
}
uretprobe:/tmp/fibonacci:main.fibonacciLoop
/@start[pid]/
{
  @ns[comm] = hist(nsecs - @start[pid]);
  delete(@start[pid]);
}'
@ns[fibonacci]:
[8K, 16K)              2 |@@@@@@@@@@@@@@@@@@@@|
[16K, 32K)             1 |@@@@@@@@@@            |
```

https://github.com/iovisor/bpftrace

# Questions?

Slides
https://github.com/florianl/talks

Gophers by
github.com/ashleymcnamara/gophers




# Resources

- BPF Performance Tools (Book)
- BPF and XDP Reference Guide